ARMADA Middleware and Communication Services

Farnam Jahanian
Real-Time Computing Laboratory
Department of EECS
University of Michigan

http://www.eecs.umich.edu/RTCL/armada/

Project Overview

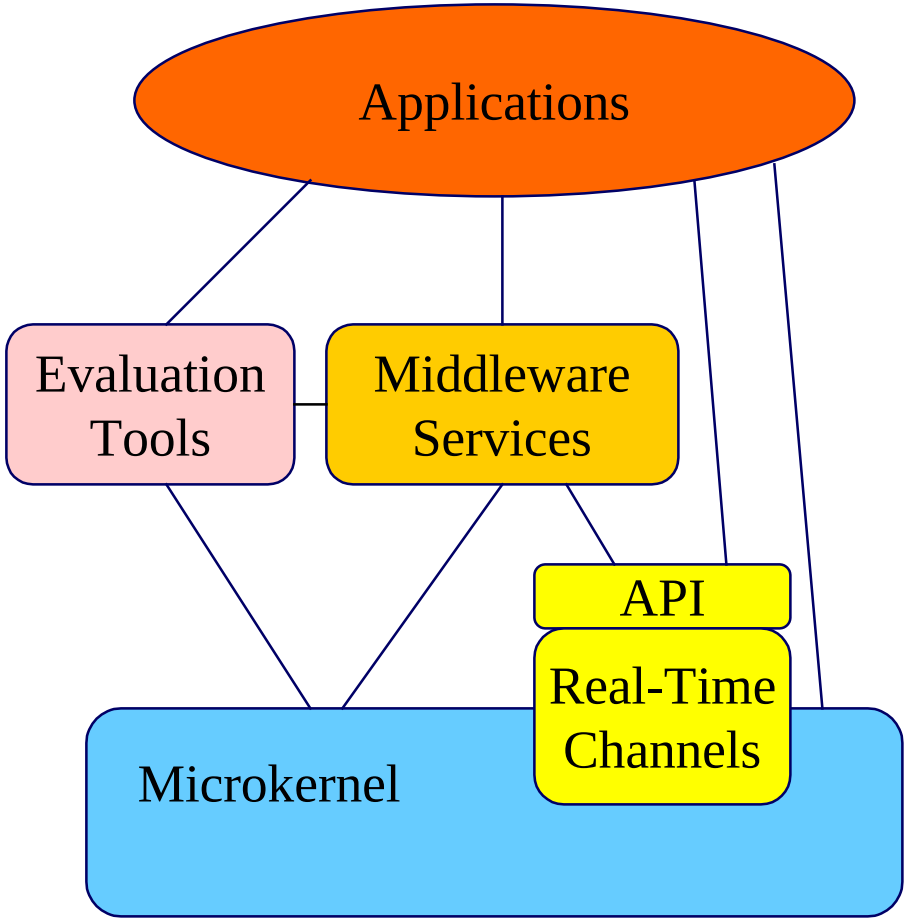
Objective:

- Design, develop, and demonstrate a software environment for building embedded real-time applications on distributed platforms.
- Joint project between University of Michigan and Honeywell Technology Center

Four complementary thrust areas:

- Real-time communication services
- Middleware Services for fault-tolerant group communication
- Dependability evaluation and validation tools
- Application demonstration

ARMADA Architecture



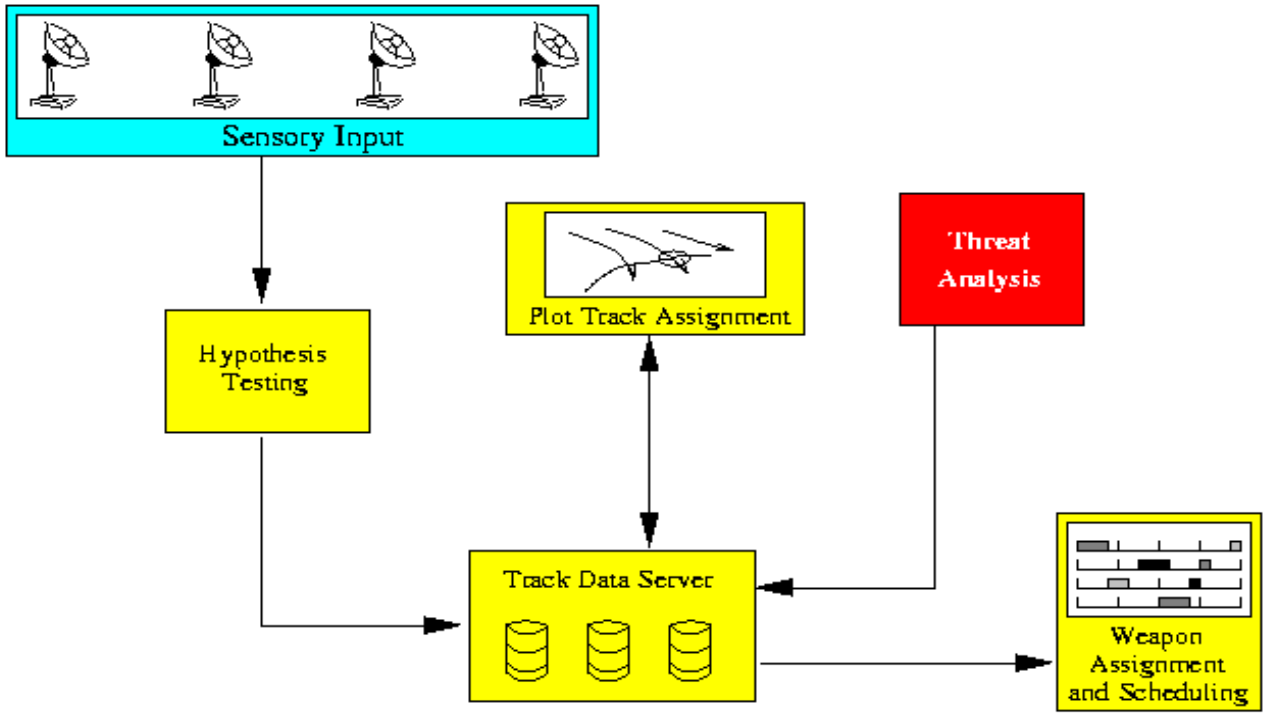
Target Applications

- Embedded fault-tolerant applications
- Industrial and manufacturing systems
- Distributed multimedia
- Air traffic control

Key Challenges:

- *Timely delivery* of services with end-to-end real-time constraints
- *Dependability* of services in the presence of h/s failures
- *Scalability* of computation and communication resources
- *Exploitation* of open systems and emerging standards in operating systems and communication services

Radar Tracking and Analysis Application



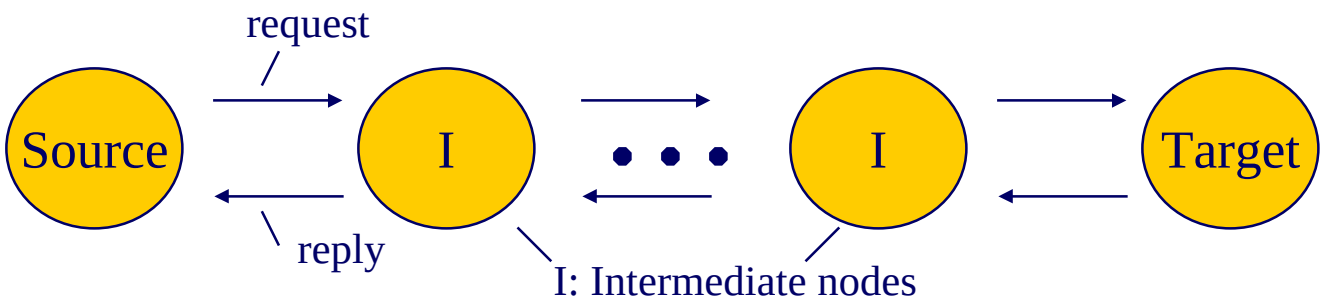
Requirements:

- End-to-end timing constraints on computation & communication tasks
- Dependability in the presence of faults

ARMADA Unicast Communication

Real-Time Channel Paradigm

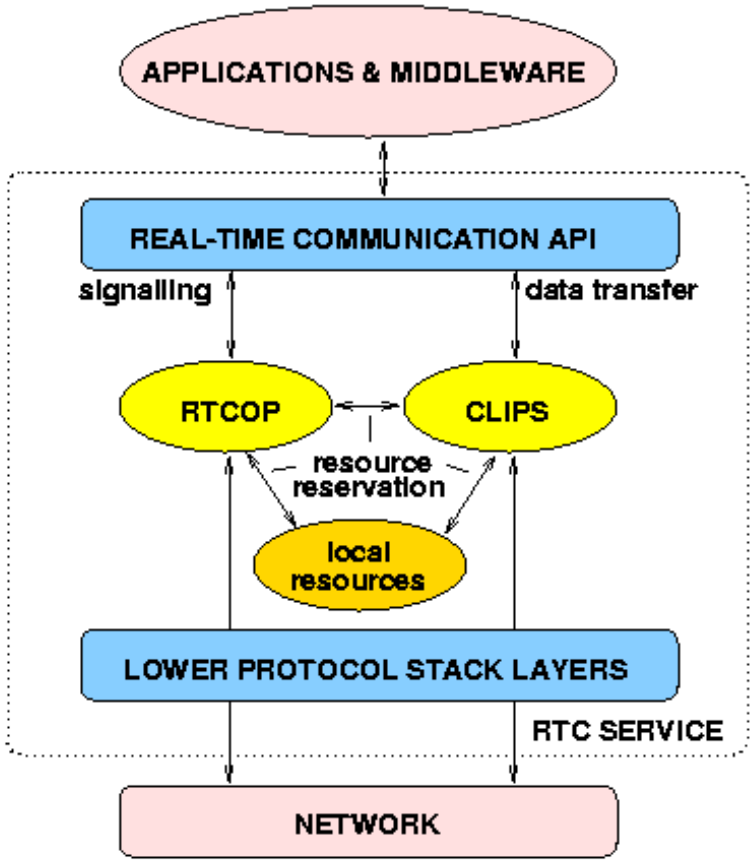
- Simplex, fixed-route, unicast virtual connection with sequenced msg
- End-to-end deterministic delay and bandwidth guarantees



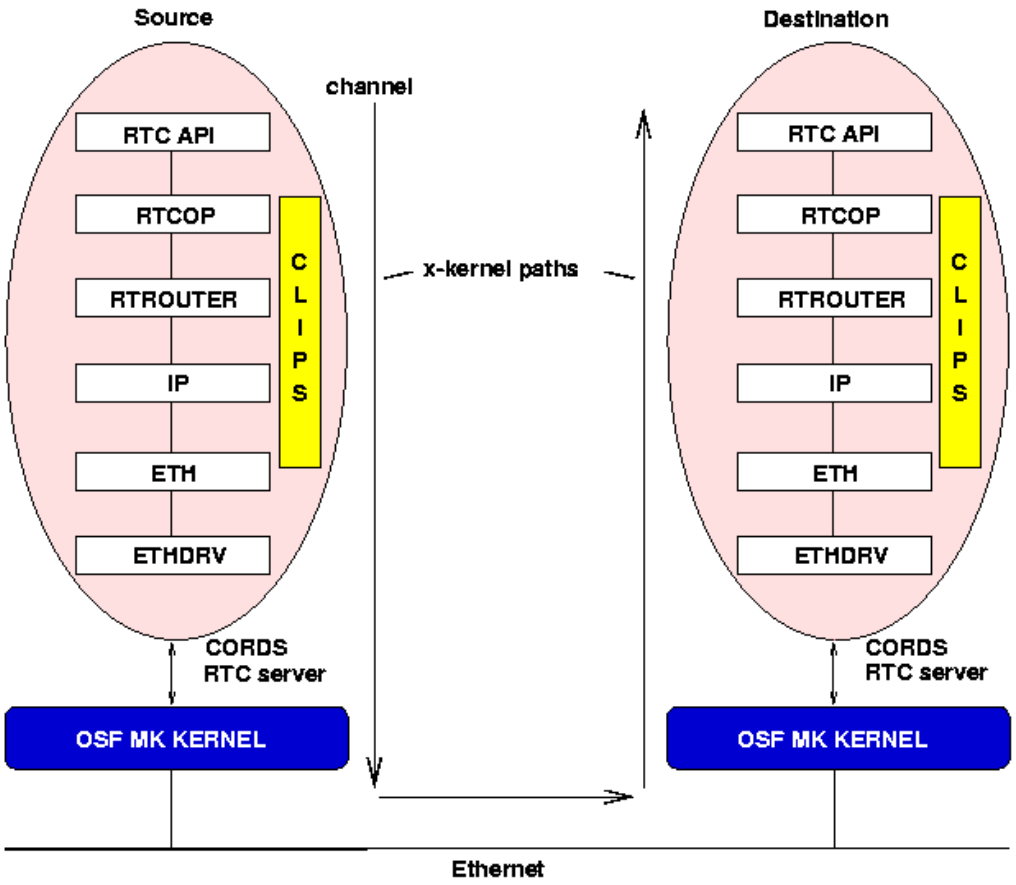
Key Features

- Three phases: channel setup, data transfer, channel teardown
- Traffic Specification: max. message size, max. burst, max. msg. rate
- QoS specification: desired worst-case end-to-end delay
- Admission control, traffic enforcement, resource scheduling

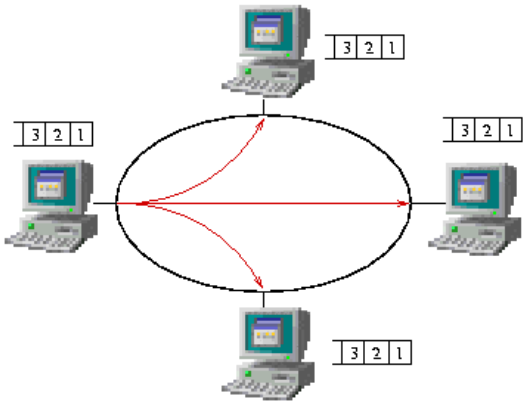
Software Architecture for Real-Time Communication



Real-Time Channel Protocol Stack



ARMADA Middleware Services

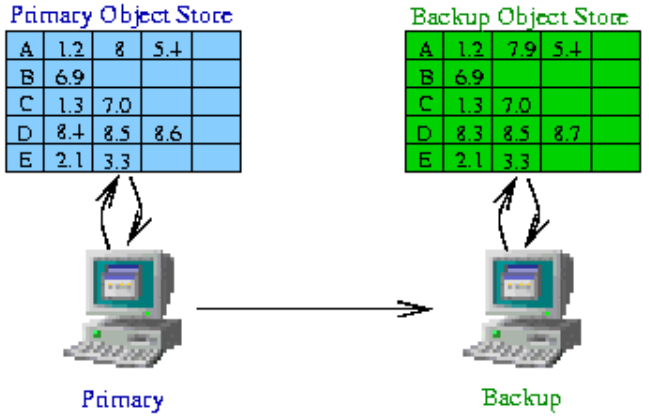


RTCAST Real-time Group Communication

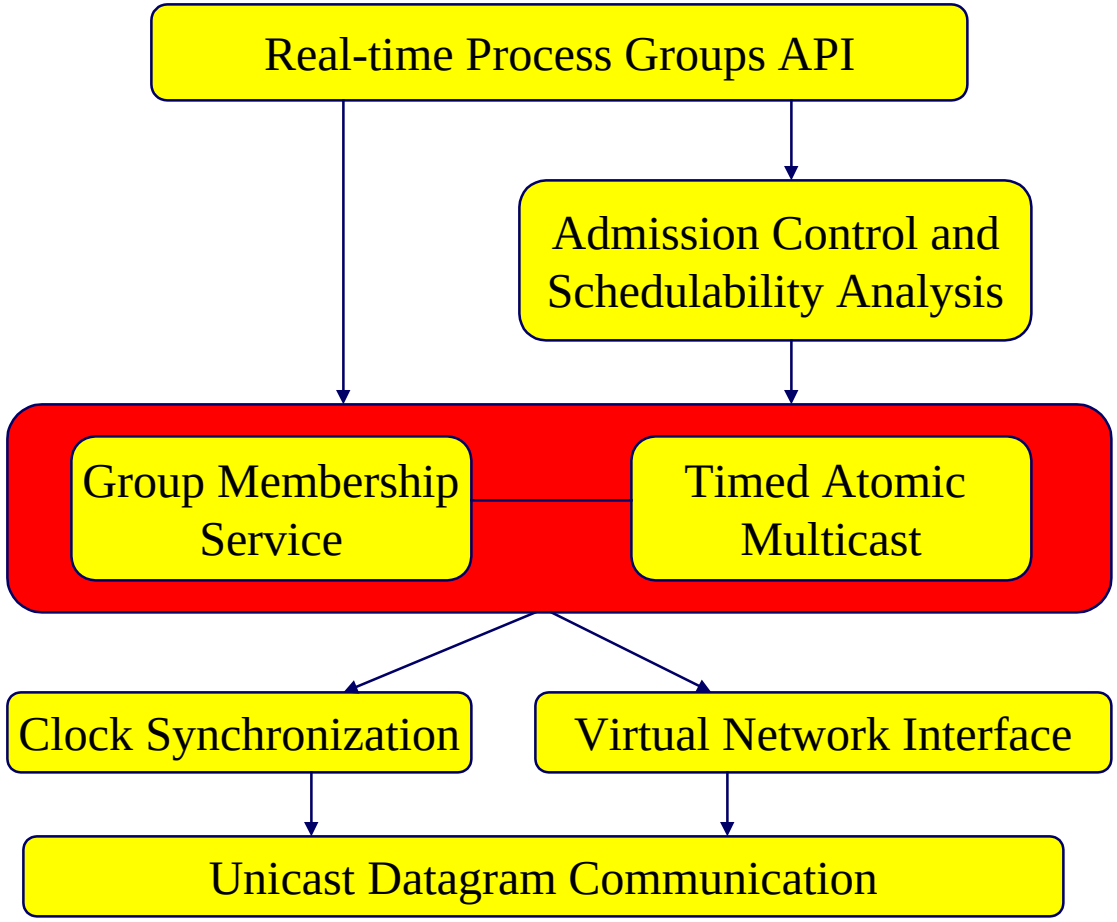
- Atomic, totally-ordered group communication
- Provides group membership service
- Supports hard or soft real-time deadlines

RTPB Real-time Primary Backup Service

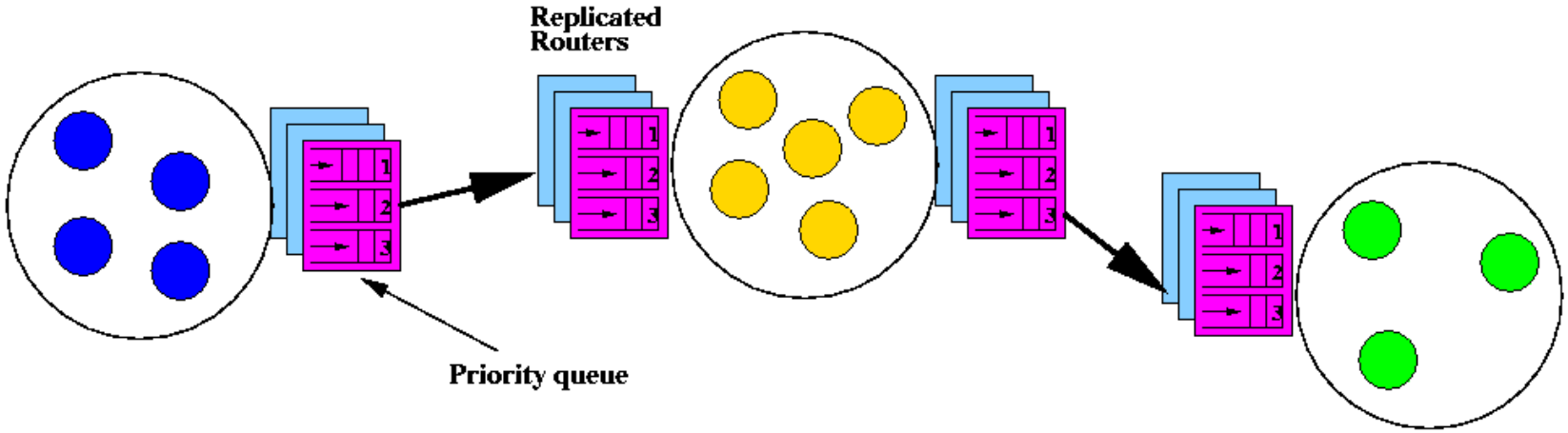
- Fault-tolerant primary-backup replication
- Supports temporal consistency between P/B
- Low overhead and fast response



The RTCAST Framework

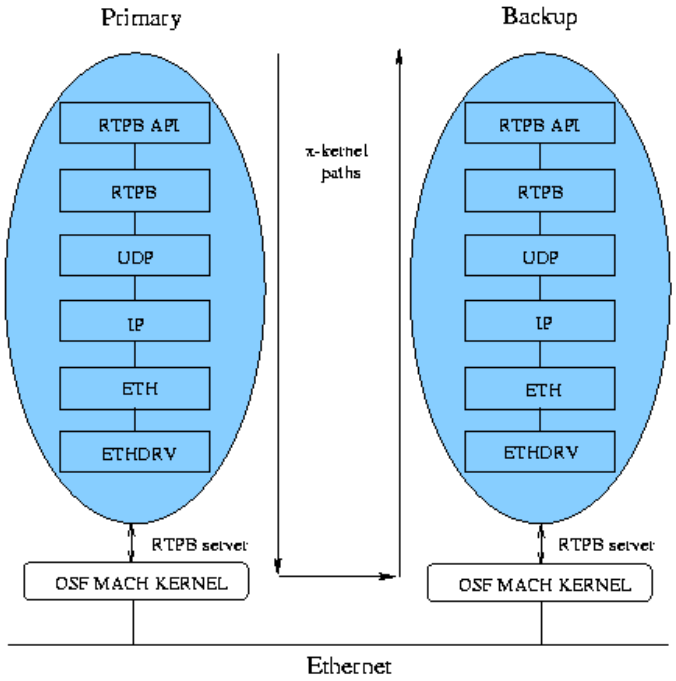


Scalable, Priority-based Inter-group Communication



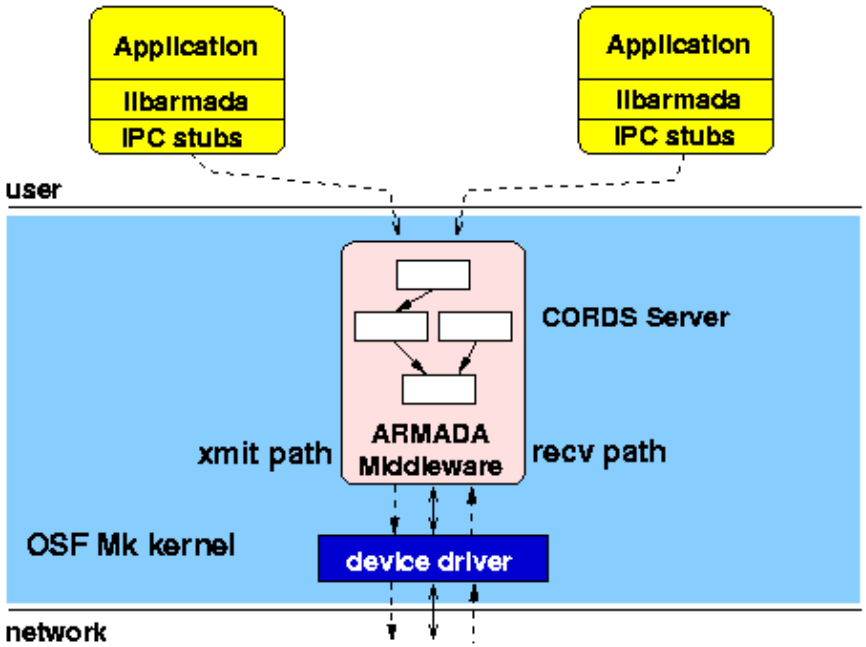
- Construct scalable distributed systems through modular group composition
- Address timeliness requirement via priority-based inter-group communication

Real-Time Primary-Backup (RTPB) Replication



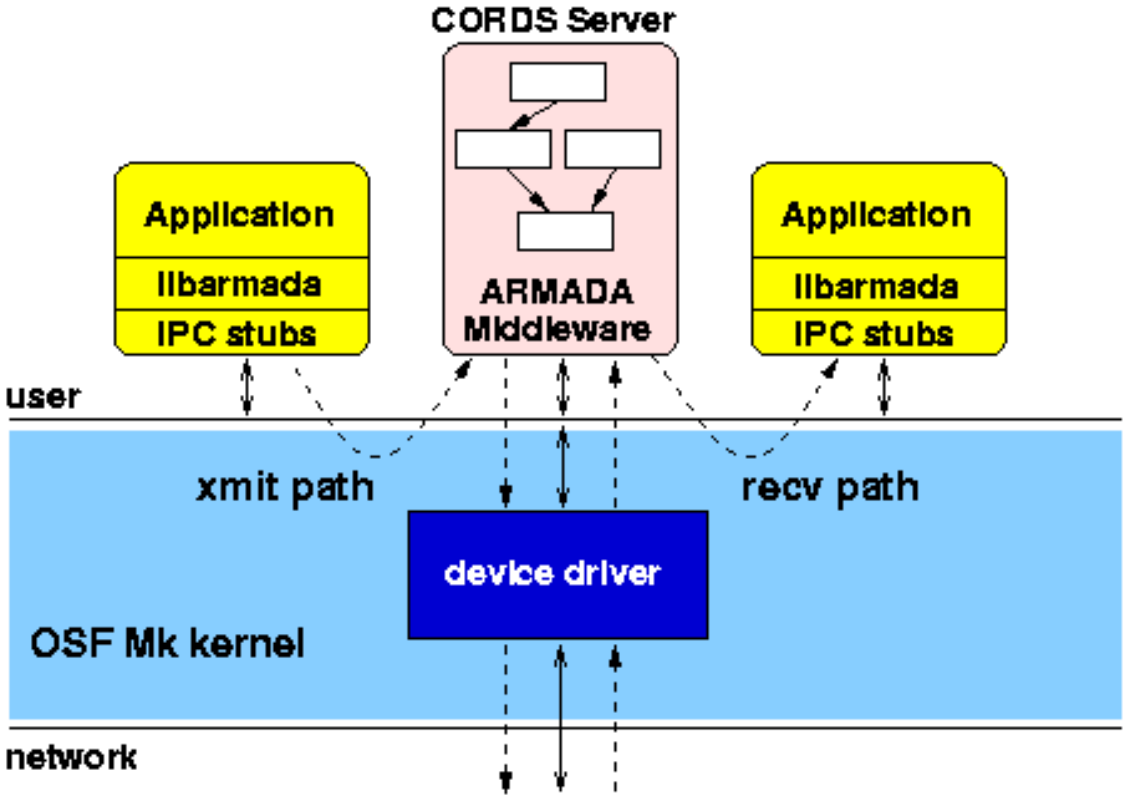
- Enforces external and inter-object consistency guarantees
- Backup is guaranteed to be consistent within a specified time bound
- Primary selectively sends updates to the backup

ARMADA Implementation Framework

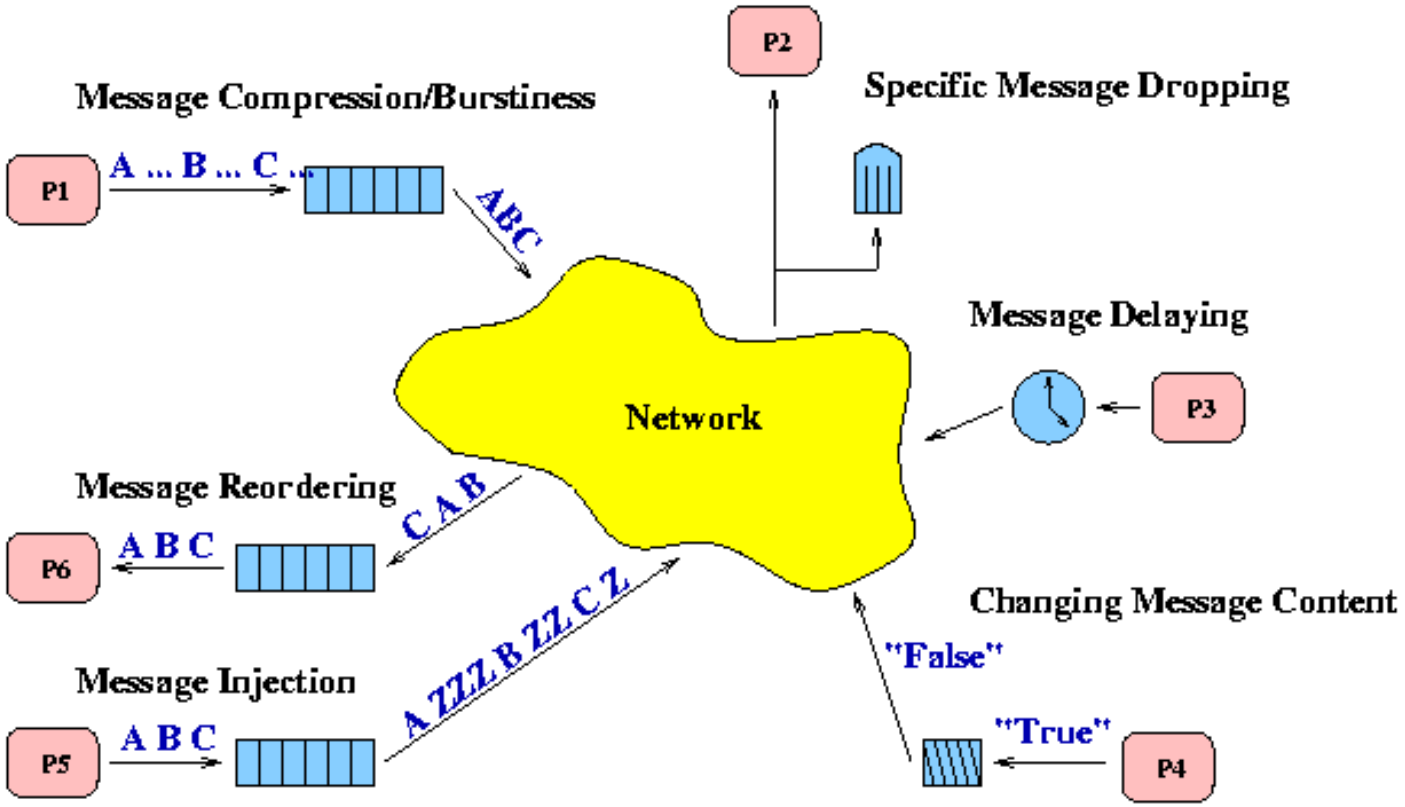


- Services are implemented in Open Group's CORDS protocol framework
- Run on the MK (RT Mach) 7.2 operating systems on Pentium PCs
- Services can be used in either user-level or in-kernel protocol stacks

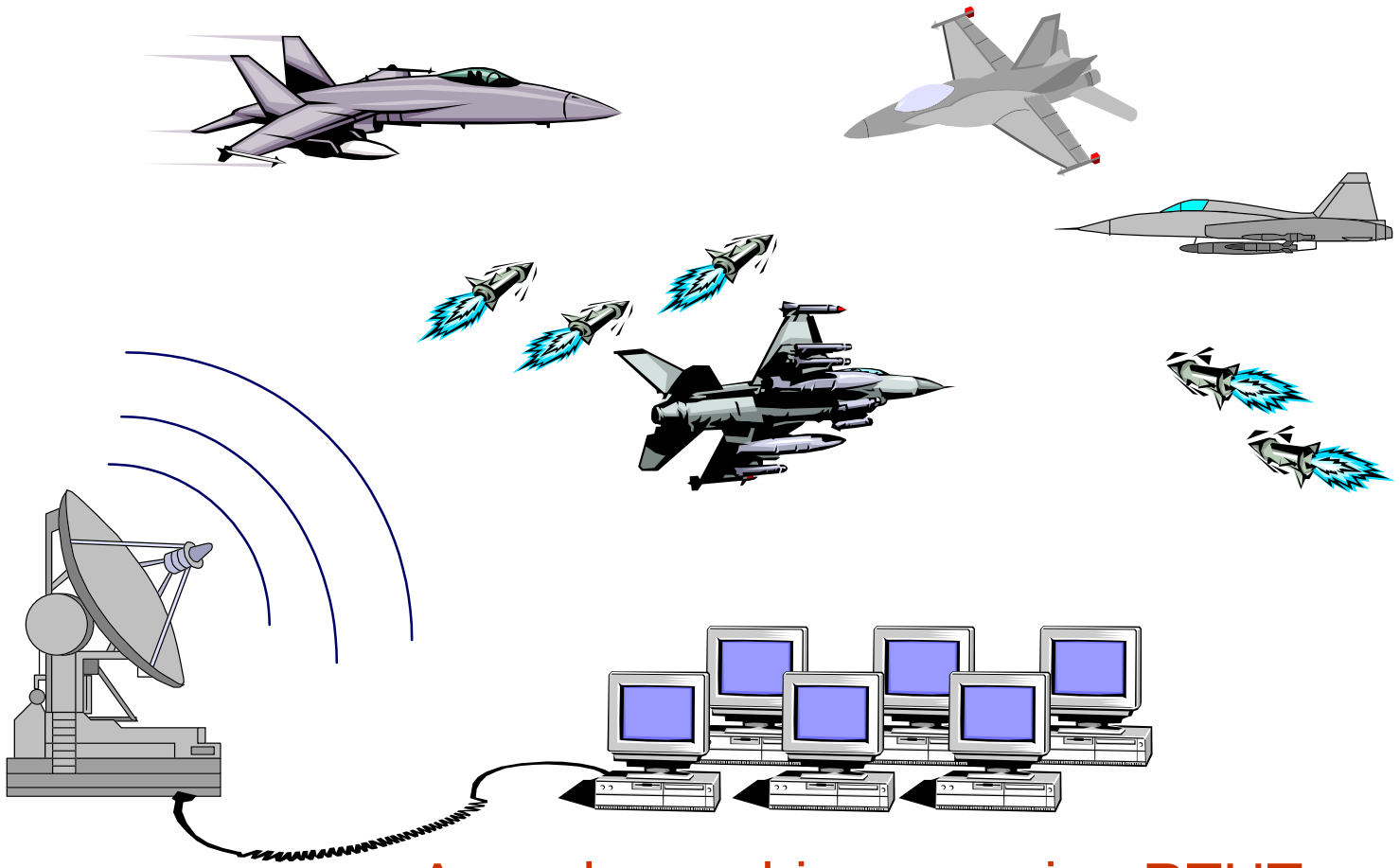
ARMADA Implementation Framework



Orchestra Fault-Injection Environment



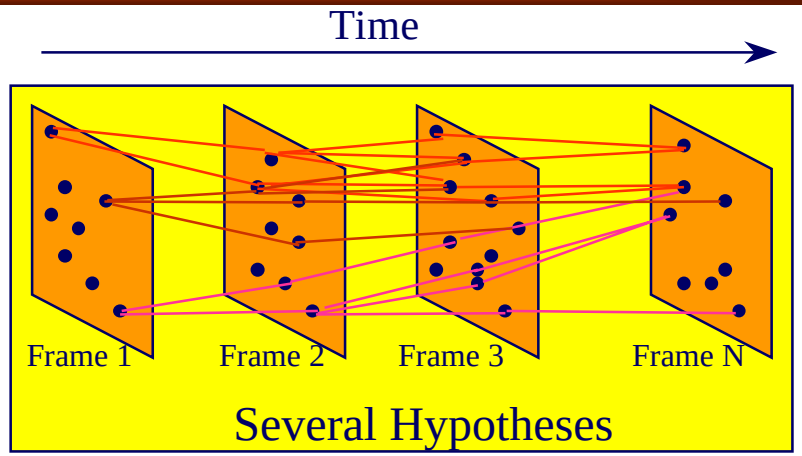
Real-Time Hypothesis Testing: A RADAR tracking application

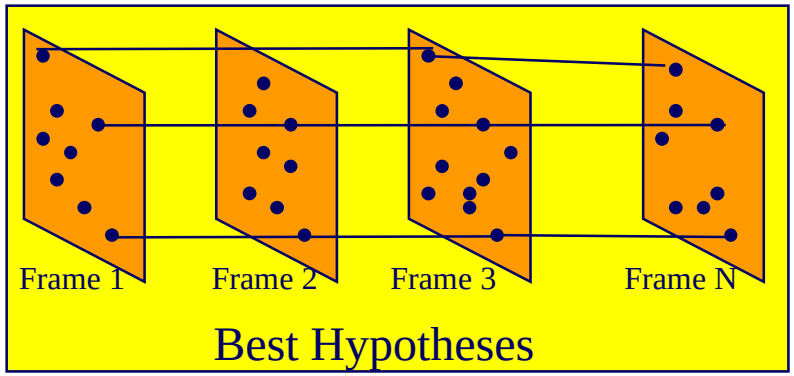


Armada machines running RTHT code

Real Time Hypothesis Testing

- Hypotheses are combinations of radar returns (from frame to frame) that could form tracks.
- There exist 100's points per frame, and 100,000's of hypotheses.
- Hypotheses are scored by their likelihood to be true. Unlikely hypotheses are pruned to avoid state explosion.



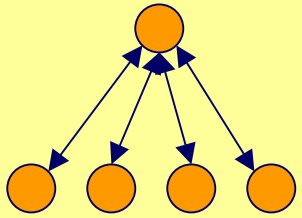


One iteration of hypothesis testing

Steps in iteration local to processor:

1. **Get new frame**
2. **extend hypothesis onto new frame**
3. **rank hypothesis globally**
4. **prune local hypothesis**

Coordinator



When ranking the best hypotheses across all processors, interprocessor communication is required with a coordinator. Otherwise, all computation is independent. How communication is handled is an important part of different fault-tolerant schemes.

Fault Tolerance in RTHT

- Each processor keeps track of a portion of the hypotheses.
- If a processor fails, its hypothesis are lost and we lose information on possible threats.
- The application must be constructed to achieve fault tolerance so that we reduce the chance of losing valuable information if a node fails.
- ARMADA calls facilitate the construction of several fault tolerance strategies.

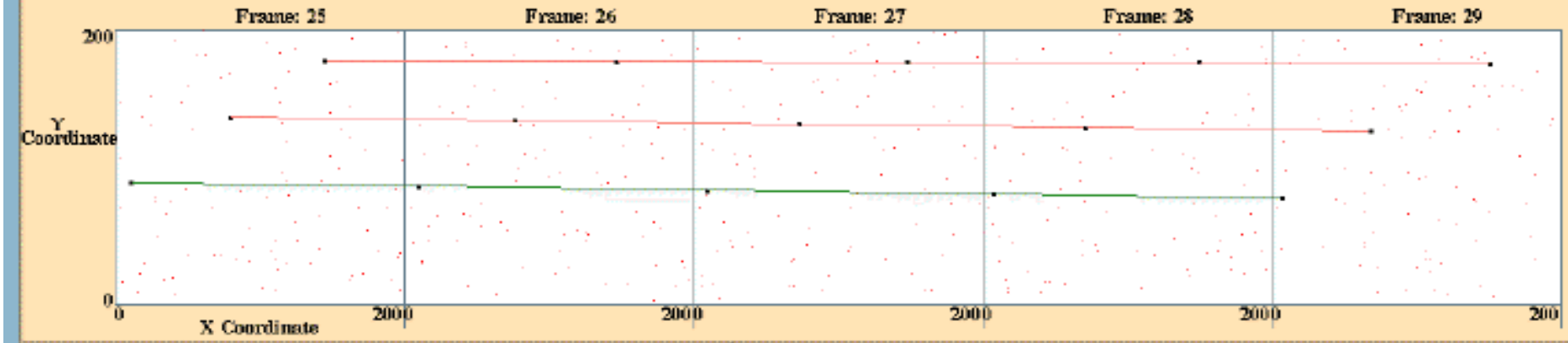
Fault Tolerance Strategies

- **Full replication of computation and data (hot spare)**
  - Coordinator aware of replication
  - Coordinator unaware of replication
- **Replication of data (not computation) on processor pairs; recovery on partner**
- **Replication of data on processor pairs with spare nodes used for recovery**

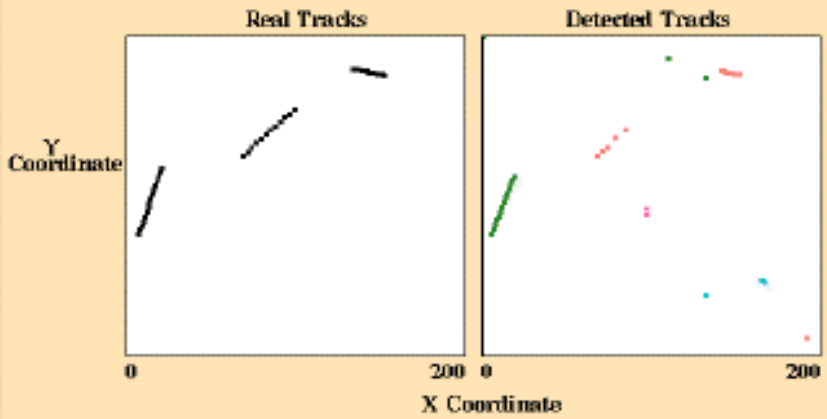
Real–Time Fault Tolerant Radar Detection

Honeywell

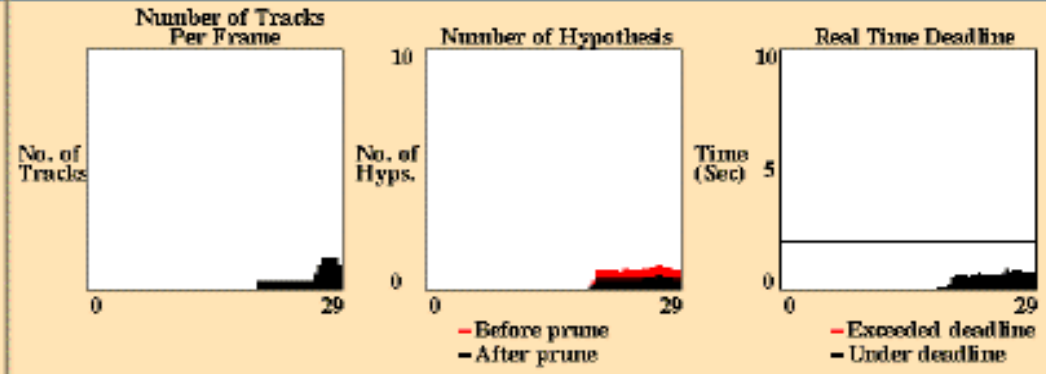
Incoming Radar Data; Tracks belong to processor of the same color.



Track Detection



Performance Monitoring



Processor IDs and Status

VCR Control Panel

Exit

Processor State Legend

Machine up

Machine down

S Spare

C Coordinator

Current Project Status

- Implementation and demonstration of real-time channels on OSF MK 7.2. Includes support for signaling and data transfer
- Detailed parameterization and profiling of RTC protocol stack
- Implementation and evaluation of RTCAST middleware
- Implementation and evaluation of RTPB middleware
- Design and Implementation of API libraries for RT Channel, RTCAST, and RTPB services
- Migration of Orchestra fault-injection tool to x-kernel MK platform. Experimental evaluation of Open Group's GIPC and ARMADA middleware
- Complete specification of radar tracking and analysis app.
- Ongoing implementation & demonstration of RTHP application on the ARMADA middleware

NT Implementation of ARMADA Middleware

Proposed JPL Interaction

1) Demonstration of ARMADA middleware to support software-implemented fault-tolerance on JPL testbed; includes active & passive replication, and time-constrained communication

2) Exploration & evaluation of ARMADA middleware on NT

3) Implementation & demonstration of a representative X2000 application on our middleware

Proposed JPL Interaction

ARMADA middleware on JPL testbed	Start date / duration: Oct.98 / one month
	Plan: remote software installation + 1 week visit
	Requirements: 4 PCs running MK and 1 SUN Solaris; technical support from JPL
ARMADA on NT demonstration & evaluation	Start date / duration: Jan. 99 / three month
	Plan: remote software installation + periodic visits
	Requirements: 4 PCs running NT and 1 SUN Solaris; technical support from JPL
X2000 application on ARMADA middleware	Start date / duration: Nov. 98 / four month
	Plan: remote software installation + extended visits
	Requirements: several PCs; specification and development of X2000 application by JPL team

Products: ARMADA middleware software / demonstration